# Comparison of Classification Models.

The objective is to compare the following classification algorithms on a particular dataset:

1. Linear Discriminant Analysis
2. Quadratic Discriminant Analysis
3. K Nearest Neighbors
4. Random Forest

We will use the caret library for this project as it's syntax is simple and straightforward.

Loading the data from the .txt file.

```
project2 = read.csv('project2.txt', header = FALSE, sep = ",")
```

Converting the response as factors so that classification models can be fit.

```
project2$V5 = as.factor(project2$V5)
```

Viewing the table

```
View(project2)
```

Viewing the dimensions of the table

```
dim(project2)
```

```
> dim(project2)
[1] 1372    5
```

Viewing the column names.

```
names(project2)
```

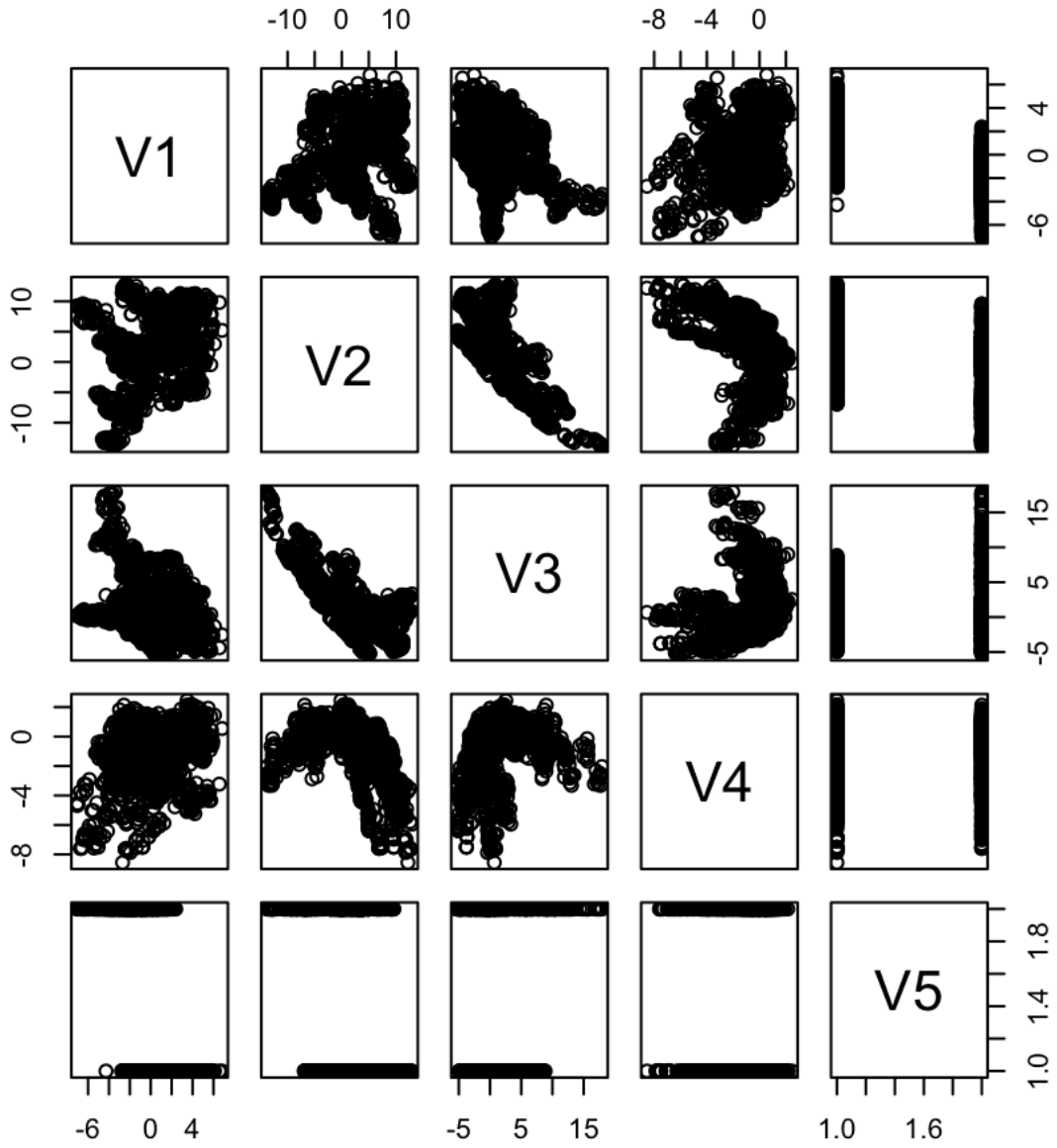
```
> names(project2)
[1] "V1" "V2" "V3" "V4" "V5"
```

Viewing the summary of the dataset.

```
summary(project2)
```

Plotting the variables

```
pairs(project2)
```




Viewing the correlation matrix

```
cor(project2[-5])
```

```
           V1         V2         V3         V4
V1  1.0000000  0.2640255 -0.3808500  0.2768167
V2  0.2640255  1.0000000 -0.7868952 -0.5263208
V3 -0.3808500 -0.7868952  1.0000000  0.3188409
V4  0.2768167 -0.5263208  0.3188409  1.0000000
```

# Linear Discriminant Analysis

First, we fit the LDA model

Splitting train and test data.

```
set.seed(12)
train.index <- createDataPartition(project2[,"V5"],p=0.8,list=FALSE)
project2.trn <- project2[train.index,]
project2.tst <- project2[-train.index,]

ctrl  <- trainControl(method  = "cv",number  = 10)
```

Fitting the model

```
fit.cv <- train(V5 ~ ., data=project2.trn, method="lda",
                trControl = ctrl)
```

Making predictions on the test data

```
pred <- predict(fit.cv,project2.tst)
```

Viewing the cross-validation results.

```
confusionMatrix(table(project2.tst[,"V5"],pred))
```

```
Confusion Matrix and Statistics

   pred
      0   1
  0 142  10
  1   0 122

               Accuracy : 0.9635
                 95% CI : (0.9339, 0.9824)
    No Information Rate : 0.5182
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9267

 Mcnemar's Test P-Value : 0.004427

            Sensitivity : 1.0000
            Specificity : 0.9242
         Pos Pred Value : 0.9342
         Neg Pred Value : 1.0000
             Prevalence : 0.5182
         Detection Rate : 0.5182
   Detection Prevalence : 0.5547
      Balanced Accuracy : 0.9621

       'Positive' Class : 0
```

Cross-Validated Accuracy for LDA is ****** 0.9635 ******
Observing the Confusion matrix, we can tell that there are ****** 10 false negatives **** out of a total of 274 predictions**
Here we assume 0 as False and 1 as True

# Quadratic Discriminant Analysis

Now we fit the QDA model

Splitting train and test data.

```
set.seed(12)
train.index <- createDataPartition(project2[,"V5"],p=0.8,list=FALSE)
project2.trn <- project2[train.index,]
project2.tst <- project2[-train.index,]

ctrl  <- trainControl(method  = "cv",number  = 10)
```

Fitting the model

```
fit.cv <- train(V5 ~ ., data=project2.trn, method="qda",
                trControl = ctrl)
```

Making predictions on the test data

```
pred <- predict(fit.cv,project2.tst)
```

Viewing the cross validation results.

```
confusionMatrix(table(project2.tst[,"V5"],pred))
```

```
Confusion Matrix and Statistics

   pred
      0   1
  0 144   8
  1   0 122

               Accuracy : 0.9708
                 95% CI : (0.9433, 0.9873)
    No Information Rate : 0.5255
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9413

 Mcnemar's Test P-Value : 0.01333

            Sensitivity : 1.0000
            Specificity : 0.9385
         Pos Pred Value : 0.9474
         Neg Pred Value : 1.0000
             Prevalence : 0.5255
         Detection Rate : 0.5255
   Detection Prevalence : 0.5547
      Balanced Accuracy : 0.9692

       'Positive' Class : 0
```

Cross-Validated Accuracy for QDA is ****** 0.9708 ******
Observing the Confusion matrix we can tell that there are ****** 8 false negatives ******
**out of a total of 274 predictions**
Here we assume 0 as False and 1 as True

# K Nearest Neighbors

Next we fit the KNN model

Splitting train and test data.

```
set.seed(12)
train.index <- createDataPartition(project2[,"V5"],p=0.8,list=FALSE)
project2.trn <- project2[train.index,]
project2.tst <- project2[-train.index,]
```

ctrl <- trainControl(method = "cv",number = 10)

Fitting the model using 10 fold CV as defined above.

```
fit.cv <- train(V5 ~ ., data = project2.trn, method = "knn",
  trControl = ctrl,
  preProcess = c("center","scale"),
  tuneGrid =data.frame(k=seq(1,35,by=1)))
```

As can be seen from the plot later, the maximum accuracy is achieved at k = 5

Making predictions on the test data

```
pred <- predict(fit.cv,project2.tst)
```

Viewing the cross validation results.

```
confusionMatrix(table(project2.tst[,"V5"],pred))
```

```
Confusion Matrix and Statistics

   pred
      0   1
  0 151   1
  1   0 122

               Accuracy : 0.9964
                 95% CI : (0.9798, 0.9999)
    No Information Rate : 0.5511
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9926

 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9919
         Pos Pred Value : 0.9934
         Neg Pred Value : 1.0000
             Prevalence : 0.5511
         Detection Rate : 0.5511
   Detection Prevalence : 0.5547
      Balanced Accuracy : 0.9959

       'Positive' Class : 0
```

Cross-Validated Accuracy for KNN is ****** 0.9964 ******
Observing the Confusion matrix we can tell that there is only ****** 1 false negatives ******
**out of a total of 274 predictions**
Here we assume 0 as False and 1 as True

The maximum accuracy is achieved at k = 5

```
print(fit.cv)
```

k-Nearest Neighbors

1098 samples
   4 predictor
   2 classes: '0', '1'

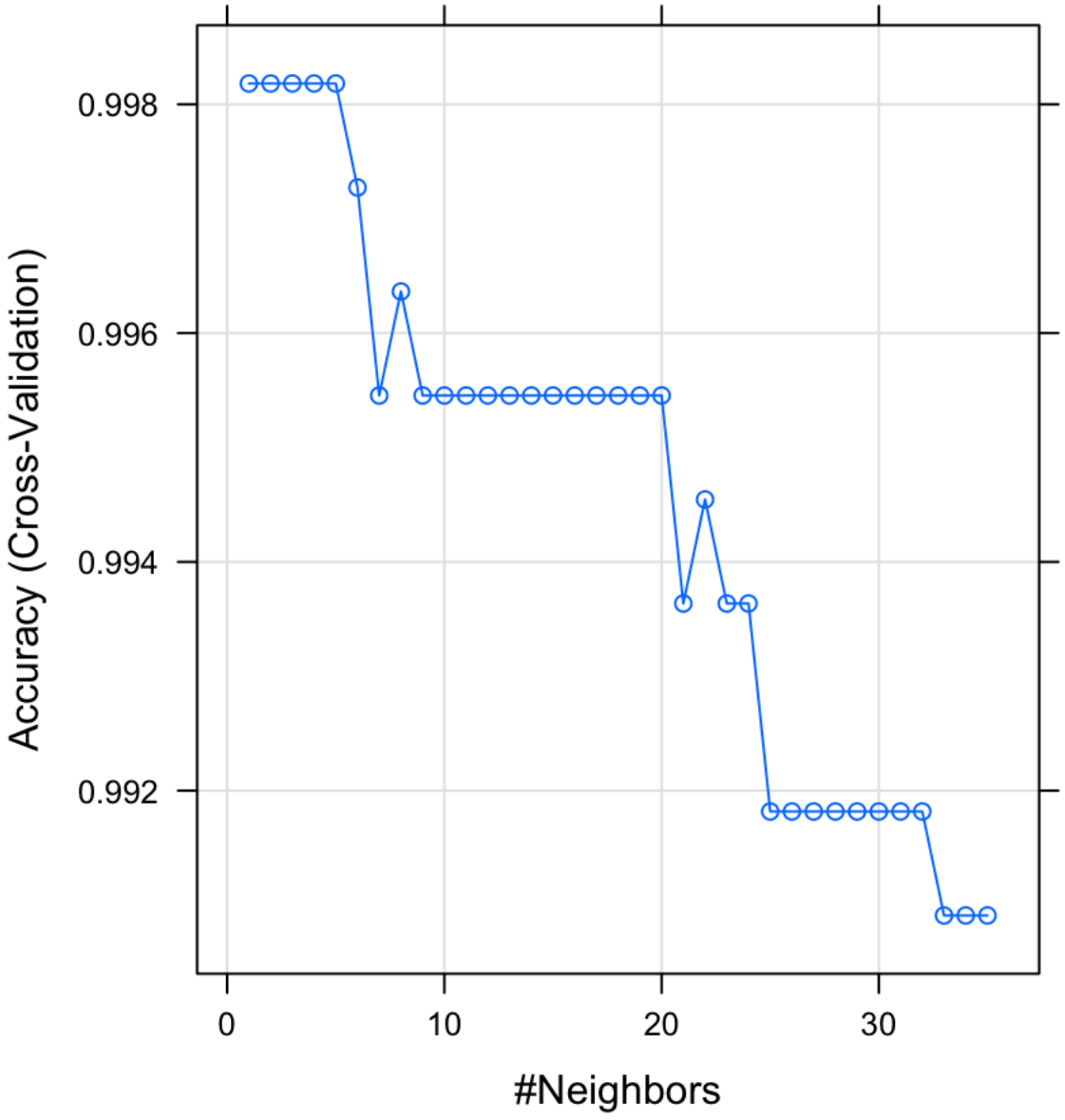
Pre-processing: centered (4), scaled (4)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 988, 988, 988, 988, 988,
Resampling results across tuning parameters:

| k | Accuracy | Kappa |
|---|---|---|
| 1 | 0.9981818 | 0.9963272 |
| 2 | 0.9981818 | 0.9963272 |
| 3 | 0.9981818 | 0.9963272 |
| 4 | 0.9981818 | 0.9963272 |
| 5 | 0.9981818 | 0.9963272 |
| 6 | 0.9972727 | 0.9944982 |
| 7 | 0.9954545 | 0.9908180 |
| 8 | 0.9963636 | 0.9926544 |
| 9 | 0.9954545 | 0.9908180 |
| 10 | 0.9954545 | 0.9908180 |
| 11 | 0.9954545 | 0.9908180 |
| 12 | 0.9954545 | 0.9908180 |
| 13 | 0.9954545 | 0.9908180 |
| 14 | 0.9954545 | 0.9908180 |
| 15 | 0.9954545 | 0.9908180 |

```
  16  0.9954545  0.9908180
  17  0.9954545  0.9908180
  18  0.9954545  0.9908180
  19  0.9954545  0.9908180
  20  0.9954545  0.9908180
  21  0.9936364  0.9871672
  22  0.9945455  0.9889890
  23  0.9936364  0.9871672
  24  0.9936364  0.9871672
  25  0.9918182  0.9835018
  26  0.9918182  0.9835018
  27  0.9918182  0.9835018
  28  0.9918182  0.9835018
  29  0.9918182  0.9835018
  30  0.9918182  0.9835018
  31  0.9918182  0.9835018
  32  0.9918182  0.9835018
  33  0.9909091  0.9816727
  34  0.9909091  0.9816727
  35  0.9909091  0.9816727

Accuracy was used to select the optimal model using
 the largest value.
The final value used for the model was k = 5.
```

The same is apparent from the plot.

```
plot(fit.cv)
```



# Random Forest

Next, we fit the Random Forest model

Splitting train and test data.

```
set.seed(0)
train.index <- createDataPartition(project2[,"V5"],p=0.8,list=FALSE)
project2.trn <- project2[train.index,]
project2.tst <- project2[-train.index,]

ctrl  <- trainControl(method  = "cv",number  = 10)
```

Fitting the model using 10-fold CV as defined above.

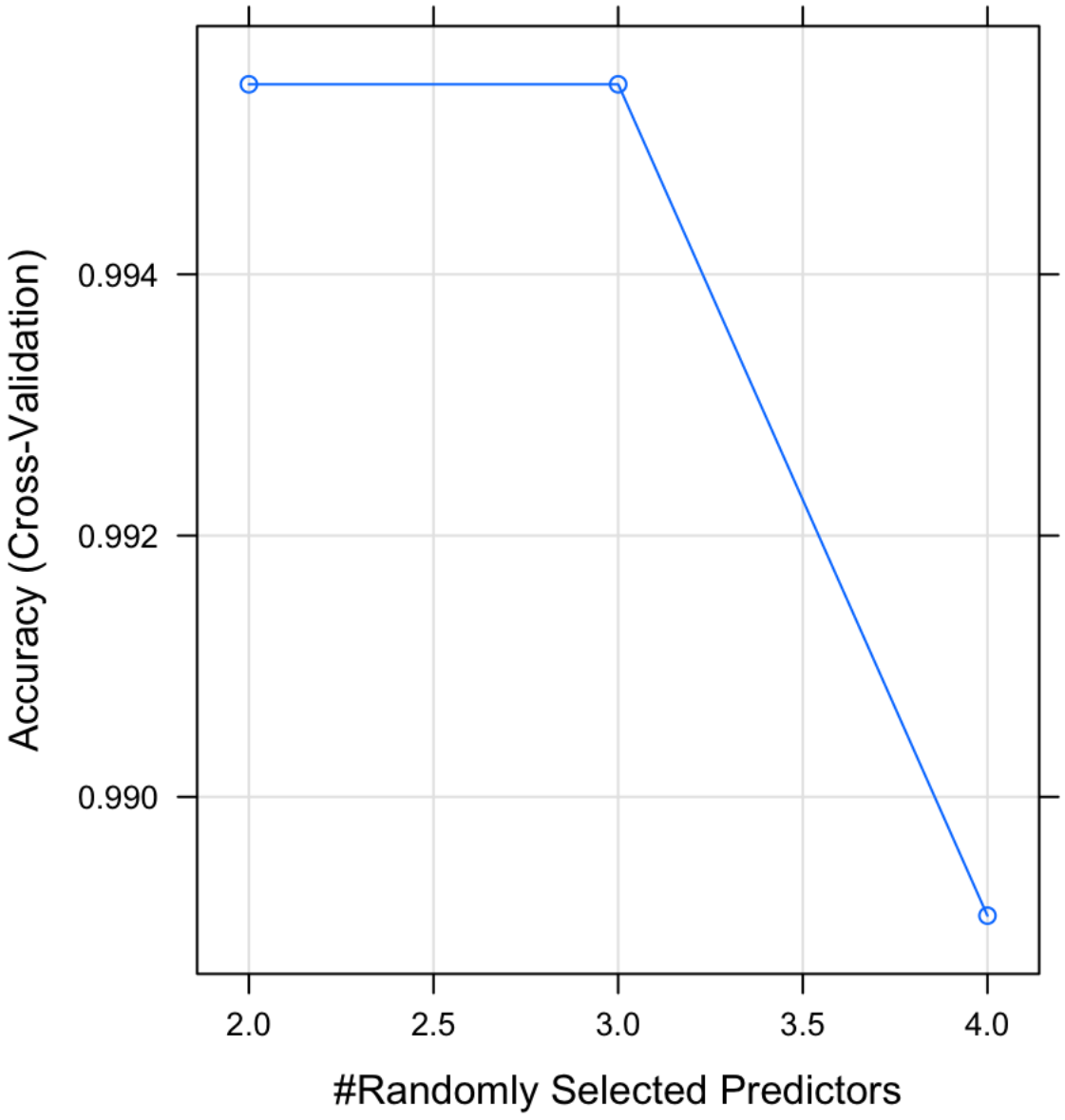
```
fit.cv <- train(V5 ~ ., data = project2.trn, method = "rf",
                trControl = ctrl,
                tuneLength = 50)
```

3 parameters were selected by the model.

```
print(fit.cv)
```

```
Random Forest

1098 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 988, 988, 989, 988, 988, 988, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.9954545  0.9908180
  3     0.9954545  0.9908033
  4     0.9890909  0.9778966

Accuracy was used to select the optimal model using
 the largest value.
The final value used for the model was mtry = 2.
```

Same is apparent from the plot.



```
  mtry  Accuracy     Kappa  AccuracySD     KappaSD
1    2 0.9954545 0.9908180 0.006428243 0.012973804
2    3 0.9954545 0.9908033 0.004791330 0.009694258
3    4 0.9890909 0.9778966 0.012712835 0.025758435
```

Accuracy is maximised with 3 variables.

Making predictions on the test data

```
pred <- predict(fit.cv,project2.tst)
```

Viewing the cross validation results.

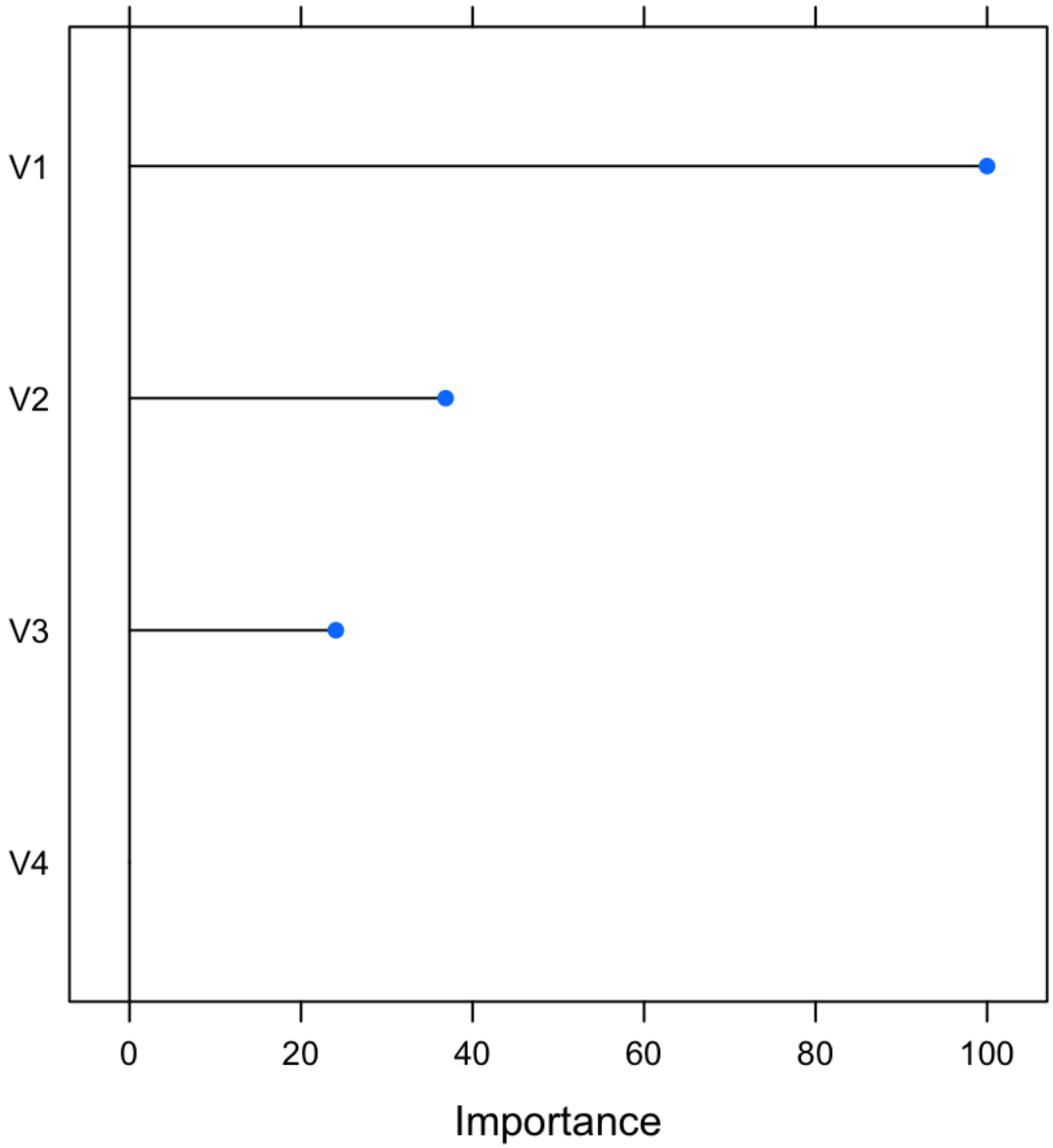
```
confusionMatrix(table(project2.tst[,"V5"],pred))
```

```
  pred
     0   1
0 149   3
1   0 122

               Accuracy : 0.9891
                 95% CI : (0.9683, 0.9977)
    No Information Rate : 0.5438
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9779

 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 1.0000
            Specificity : 0.9760
         Pos Pred Value : 0.9803
         Neg Pred Value : 1.0000
             Prevalence : 0.5438
         Detection Rate : 0.5438
   Detection Prevalence : 0.5547
      Balanced Accuracy : 0.9880

       'Positive' Class : 0
```

Cross-Validated Accuracy for KNN is ****** 0.9891 ******
Observing the Confusion matrix we can tell that there are ****** 3 false negatives ******
**out of a total of 274 predictions**
Here we assume 0 as False and 1 as True

Let us now check the variable importance.
As can be seen from the plot and the table, V1 is the most important variable in the dataset.

```
print(varImp(fit.cv)) # Variable importance
plot(varImp(fit.cv))
```



```
rf variable importance

   Overall
V1  100.00
V2   36.87
V3   24.07
V4    0.00
```

# Conclusion

The following table summarizes the findings:

| Model | LDA | QDA | KNN | RF |
|---|---|---|---|---|
| Cross Validated Accuracy | **0.9635** | **0.9708** | **0.9964** | **0.9891** |
| Confusion Matrix | **0** 142 10<br>**1** 0 122 | **0** 144 8<br>**1** 0 122 | **0** 151 1<br>**1** 0 122 | **0** 149 3<br>**1** 0 122 |

In conclusion, KNN appears to produce the best results with 99.64% Accuracy.